NLP Project for Multi Class Text Classification using BERT Model
Project Overview

Business Overview

So far, in this series of NLP projects for our multiclass text classification problem, we have come across several algorithms such as; the Naïve Bayes algorithm, skip-gram model, Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) as well as attention mechanism. All these models were built and implemented from scratch.

In this particular project, we will be using a pre-trained model to predict our text known as BERT. BERT is an open-source ML framework for Natural Language Processing. BERT stands for Bidirectional Encoder Representations and is a pre-trained model from Google known for producing state-of-the-art results in a wide variety of NLP tasks.

Aim

To perform multiclass text classification on the dataset using the pre-trained BERT model.

Data Description

The dataset contains more than two million customer complaints about consumer financial products. Amongst the various available columns, we have a column that contains the actual text of the complaint and one column containing the product for which the customer is raising the complaint.

Tech Stack

- ➔ Language: Python
- ➔ Libraries: pandas, torch, nltk, numpy, pickle, re, tqdm, sklearn, transformers

Prerequisite

1. The torch framework
2. Multiclass Text Classification using Naive Bayes in Python
3. Skip Gram Model Python Implementation for Word Embeddings
4. Build Multi Class Text Classification Models with RNN and LSTM
5. Build a Text Classification Model with Attention Mechanism NLP

Approach

1. Installing the necessary packages through the pip command
2. Importing the required libraries
3. Defining configuration file paths
4. Process Text data
    - Read the CSV file and drop the null values
    - Replace duplicate labels
    - Encode the label column and save the encoder and encoded labels
5. Data Preprocessing
    - Conversion to lower case
    - Punctuation removal
    - Digits removal
    - Remove more than one consecutive instance of 'x'
    - Additional spaces removal
    - Tokenize the text
    - Save the tokens
6. Model Building
    - Create BERT model
    - Create a function for the PyTorch dataset
    - Function to train the model
    - Function to test the model
7. Train BERT model
    - Load the files
    - Split data into train, test, and validation
    - Create PyTorch datasets
    - Create data loaders
    - Create model object
    - Define loss function and optimizer
    - Move the model to GPU if available

    - Training the model
    - Test the model
8. Predictions of new text

Modular Code Overview

```
Input
  |_complaints.csv


Output
  |_bert_pre_trained.pth
  |_label_encoder.pkl
  |_labels.pkl
  |_tokens.pkl


Source
  |_data.py
  |_model.py
  |_utils.py


|_bert.ipynb
|_config.py
|_Engine.py
|_predict.py
|_processing.py
|_README.MD
|_requirements.txt
```

Once you unzip the modular_code.zip file, you can find the following folders within it.

1. Input
2. Output
3. Source

1. The input folder contains the data that we have for analysis. In our case, it
    - complaints.csv

2. The source folder contains all the modularized code for all the above steps in a modularized manner. It includes the following.
    - model.py
    - data.py
    - utils.py

    These all-python files contain helpful functions which are being used in the Engine.py file.

3. The output folder contains the following;
    - bert_pre_trained.pth
    - label_encoder.pkl
    - labels.pkl
    - tokens.pkl

    These are required for our model training and can be quickly loaded and used for future use, and the user need not have to train all the models from the beginning.

4. The config.py file contains all the configurations required for this project.

5. The Engine.py file is the main file that needs to be called to run the entire code in one go. It trains the model and saves it in the output folder.
   Note: Please check the README.md file for more information.

6. The bert.ipynb is the original notebook we saw in the videos.

7. The processing.py file is used to process the data.

8. The predict.py file is used to make predictions on the data.

9. The README.md file contains all the information on how to run particular files and more instructions to follow.

10. The requirements.txt file has all the required libraries with respective versions. Kindly install the file by using the command **pip install -r requirements.txt**

Project Takeaways

1. Understanding the business problem
2. What are pre-trained models?
3. Introduction to the BERT model
4. Understanding the working BERT
5. Data preparation for the model
6. How to remove spaces and digits?
7. Removing the punctuations
8. How to perform BERT tokenization?
9. How to create the architecture for the BERT model?
10. How to create a data loader for the BERT model?
11. Building the BERT model
12. Training the pre-trained BERT model using CUDA or CPU
13. How to make predictions on the new text data?